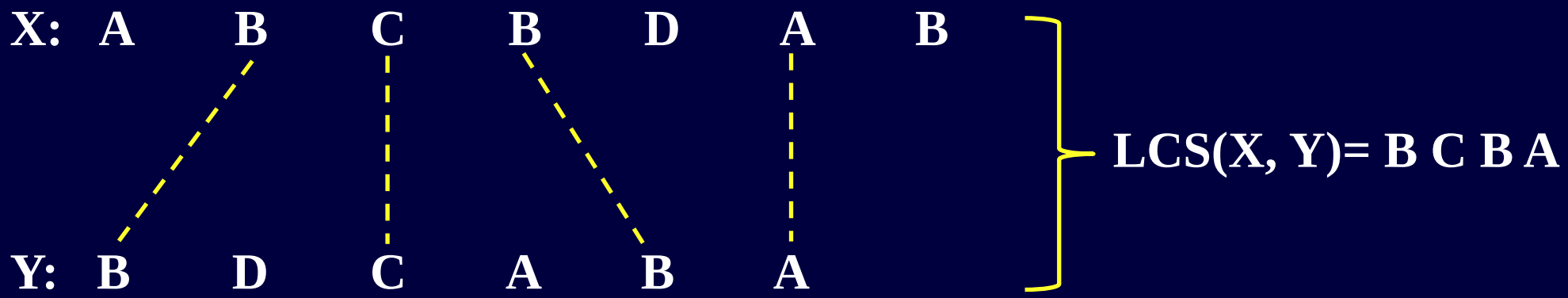
最长公共子序列

最长公共子序列(LCS)

- **最长公共子序列 Longest Common Subsequence (LCS)**

  ——给定两个序列X和Y，找到它们的公共子序列中最长的一条。



序列{B,D,A,B}同样是序列X和Y的最长公共子序列，长度为4。

最长公共子序列可以有多个，但最长公共子序列的长度是唯一的。

最长公共子序列(LCS)

子序列

B是A的子列，若B是A中删去某些元素(亦可不删)所得的序列。

即B不一定是由A的连续元素构成的子列。

定义

给定序列$X =< x_1, x_2, \dots, x_m >$，序列$Z =< z_1, z_2, \dots, z_k >$是$X$的一个子序列须满足：若$X$的索引中存在一个严格增的序列$i_1 < i_2 < \cdots < i_k$，使得对所有的$j(1 \le j \le k)$，均有$X_{i_j} = Z_j$成立。

最长公共子序列(LCS)

- 两个序列的公共子序列(CS)

  若Z是X的子列，又是Y的子列，则Z是序列X和Y的公共子序列

- 最长公共子序列(LCS)

  X和Y的公共子序列中长度最大者

- 如何求解两个给定序列的LCS
  - Step1：刻画LCS结构特征
  - Step2：子问题的递归解
  - Step3：计算最优解
  - Step4：构造一个LCS

如何求两给定序列的LCS

Step1：刻划LCS结构特征

❖穷举无效

枚举X的所有子列，检查其是否亦为Y的子列

若$|X| = m$ ，则子序列数为$2^m - 1$

❖LCS问题有一个最优子结构特性

定义序列X的前i个元素构成的**前缀**：

$X_i =< x_1, x_2, \dots, x_i >, 1 \le i \le m, x_0 = \Phi$（空串）

如何求两给定序列的LCS

一个LCS的最优子结构

设$X =< x_1, x_2, \dots, x_m >$和$Y =< y_1, y_2, \dots, y_n >$是序列，

$Z =< z_1, z_2, \dots, z_k >$是$X$和$Y$的一个任意LCS。

(1) 若$x_m = y_n$，则 $z_k = x_m = y_n$ 且 Z_{k-1}是X_{m-1}和Y_{n-1}的一个LCS。 Case 1

(2) 若$x_m \neq y_n$，且 $z_k \neq x_m$，则Z是X_{m-1}和Y的一个LCS。

(3) 若$x_m \neq y_n$，且 $z_k \neq y_n$，则Z是X和Y_{n-1}的一个LCS。

} Case 2

pf（反证法）：

（1）若$z_k \neq x_m$（则$z_k \neq y_n$），则将x_m加入Z尾，于是获得X和Y的长度为$k+1$的CS，这与Z是X和Y的LCS矛盾！

$\because z_k = x_m = y_n$

$\therefore$ 前缀Z_{k-1}是X_{m-1}和Y_{n-1}的公共子序列（长度为$k-1$）

若Z_{k-1}不是X_{m-1}和Y_{n-1}的LCS，则存在一个X_{m-1}和Y_{n-1}的公共子序列W，其长度大于$k-1$，于是将z_k附加于W，则产生的公共子序列长度大于k，这与Z是X和Y的LCS（长度为k）矛盾！

(2) 若$x_m \neq y_n$，且$z_k \neq x_m$，则Z是X_{m-1}和Y的一个LCS

若$z_k \neq x_m$， z_k必等于x_m之前的某个元素，则Z是X_{m-1}和Y的一个CS

现需证Z是一个LCS

若Z不是X_{m-1}和Y的LCS

则存在一个长度$> k$的子序列W，它显然也是X和Y的CS，这与Z是X和Y的LCS矛盾！

(3) 与(2) 对称

该定理表示，LCS问题有“**最优子结构**”性质：

动态规划特性#1（最优子结构）：一个问题的最优解也包含了子问题的最优解

两个序列的一个LCS包含了两个序列的前缀子序列的一个LCS

蕴含的选择：当$x_m \neq y_n$时，我们事先并不知道Z的长度，只能是在X_{m-1}和Y的LCS，以及在X和Y_{n-1}的LCS中取最大者。

如何求两给定序列的LCS

Step2：子问题的递归解

定理： 找X和Y的一个LCS，可分解为1个(或2个)子问题。

(1) $If\ \ x_m = y_n\ \ then$ 需解1个子问题：找X_{m-1}和Y_{n-1}的1个LCS。

(2) $If\ \ x_m \neq y_n\ \ then$ 需解2个子问题：找X_{m-1}和Y的1个LCS，找X和Y_{n-1}的1个LCS

最长者为X和Y的LCS。

LCS里重叠子问题特性：

找X和Y的1个LCS → 找$X_{m-1}\&Y$，以及$X\&Y_{n-1}$的LCS。

它们都包含子子问题：找X_{m-1}和Y_{n-1}的1个LCS，还有许多其它共享的子子问题。

如何求两给定序列的LCS

- 用C记录最优解的值

$C[i,j]$定义为X_i和Y_j的一个LCS的长度，$0 \le i \le m, 0 \le j \le n$

$$C[i,j] = \begin{cases} 0 & if\ \ i=0\ \ or\ \ j=0 \\ C[i-1,j-1]+1 & if\ \ i,j>0, x_i = y_j \\ max\{C[i,j-1], C[i-1,j]\} & if\ \ i,j>0, x_i \neq y_j \end{cases}$$

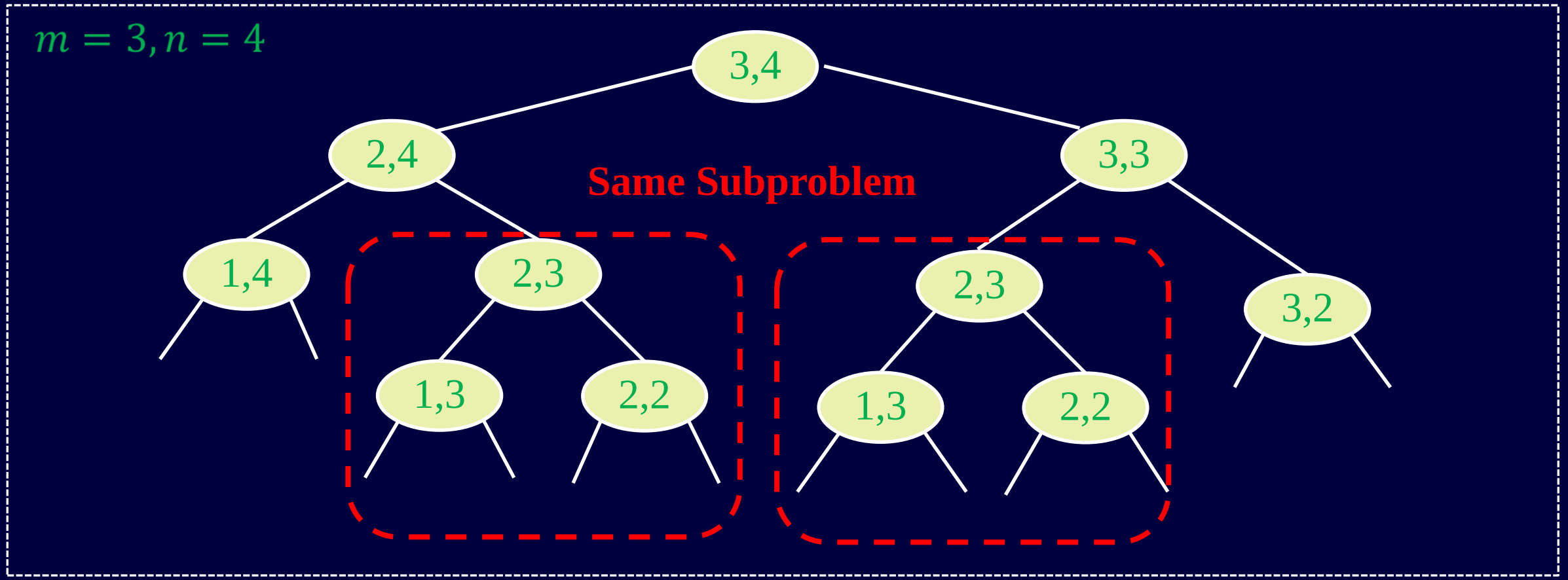
至此，可以给出直接递归版本的实现

为何引入第0行和第0列？

因为$C[i,j]$是当前行和前一行元素构成或由当前列和前一列元素构成，

故可起哨兵作用。

平凡递归算法对应的递归树



动态规划的特性#2（重叠子问题）

一个递归解中包含少量的独立子问题，但这些独立子问题会被重复计算多次。

2个序列的LCS问题产生的子问题数为mn个

如何求两给定序列的LCS

- **Step3：计算最优解**
  - 由递归关系易写自然递归算法(指数阶)，但子问题空间规模$O(mn)$.
  - 输入：$X =< x_1, x_2, \dots, x_m >$, $Y =< y_1, y_2, \dots, y_n >$
  - 输出：$C[0..m, 0..n]$ ——计算次序行主序

    $b[1..m, 1..n]$ ——解矩阵(空串无需表示出解)

$$b[i,j] = \begin{cases} \nwarrow & if \quad C[i,j]\text{由}C[i-1,j-1]\text{确定} \\ \uparrow & if \quad C[i,j]\text{由}C[i-1,j]\text{确定} \\ \leftarrow & if \quad C[i,j]\text{由}C[i,j-1]\text{确定} \end{cases}$$

  - 故当构造解时，从$b[m,n]$出发，上溯至$i=0$或$j=0$为止，当$b[i,j]$包含“↖”时打印出x_i即可。

```
LCS_LENGTH(X, Y)
    m = X.length
    n = Y.length
    for i = 1 to m
        C[i, 0] = 0                 //第0列
    for j = 1 to n
        C[0, j] = 0                 //第0行
    for i = 1 to m                  //依次考虑X_1, X_2, ..., X_m的前缀子序列
        for j = 1 to n
            if x_i = y_j
                C[i, j] = C[i − 1, j − 1] + 1
                b[i, j] = " ↖ "
            elseif C[i − 1, j] ≥ C[i, j − 1]
                C[i, j] = C[i − 1, j]       //前一行，X_{i−1}和Y_j决定
                b[i, j] = " ↑ "
            else
                C[i, j] = C[i, j − 1]       //前一列， X_i和Y_{j−1}决定
                b[i, j] = " ← "
    return C and b                  //时间Θ(mn)，空间Θ(mn)
```

如何求两给定序列的LCS

- **Step4：构造一个LCS**
  - 初始调用$PRINT_LCS(b, X, length[X], length[Y])$，当$b[i,j]=$“↖”时，有$x_i = y_j$，打印之。从$b[m,n]$开始据箭头回溯至$i=0$或$j=0$即可。
  - 用循环则打印元素为逆序，可用递归算法使之逆置

```
PRINT_LCS(b, X, i, j)        //时间O(m+n)
    if i = 0 or j = 0
        return
    if b[i,j] == "↖"
        PRINT_LCS(b, X, i, j)
        print(x_i)           //相当于后序遍历
    elseif b[i,j] == "↑"
        PRINT_LCS(b, X, i-1, j)
    else PRINT_LCS(b, X, i, j-1)
```

最长公共子序列(LCS)

改进代码

时空常数因子改变，但时间渐近性能不变

- 可省略b

  $\because C[i,j]$来自于$C[i-1,j-1], C[i-1,j], C[i,j-1]$

  $\therefore$可在$O(1)$判定是由谁计算产生的，故可在$O(m+n)$重构LCS。

  求解过程时间不变，空间减少。

- C只需要两行：当前行和前一行

  无法构造出解，但空间由$O(mn)$变为$O(m+n)$.

填表求解LCS

		A	B	C	B	D	A	B
	0	0	0	0	0	0	0	0
B	0	0	1	1	1	1	1	1
D	0	0	1	1	1	2	2	2
C	0	0	1	2	2	2	2	2
A	0	1	1	2	2	2	3	3
B	0	1	2	2	3	3	3	4
A	0	1	2	2	3	3	4	4